Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 97 (2016) 122 – 125

Procedia Computer Science

CLOUD FORWARD: From Distributed to Complete Computing, CF2016, 18-20 October 2016, Madrid, Spain

A Context Model and Policies Management Framework for Reconfigurable-by-design Distributed Applications

Panagiotis Gouvas[a], Eleni Fotopoulou[a], Anastasios Zafeiropoulos[a,*], Constantinos Vassilakis[a]

[a]*Ubitech Ltd., R&D Department, Athens, Greece*

Abstract

The design and development of distributed applications consisting of microservices is an emerging pattern, considering the advantages associated with the management of self-deployable and orchestratable components as well as the adoption of a DevOps culture in cloud applications deployment and management. In this position paper, we present a context model for representing the entire lifecycle of reconfigurable-by-design distributed applications consisting of microservices and denoted in the form of a service graph. Based on the context model, we describe a policies management framework targeted at service providers for managing deployment and orchestration aspects of such applications over a programmable infrastructure.

Peer-review under responsibility of organizing committee of the international conference on cloud forward: From Distributed to Complete Computing

Keywords: reconfigurable by design applications; microservices; context model; policies management; distributed applications orchestration

1. Introduction

Application requirements have changed dramatically in recent years raising new demands for software architectures. Such demands include the need for supporting horizontal scalability, elasticity, adaptability, resiliency and fault-tolerance characteristics, as noted in the Reactive Manifesto[1]. The design of reactive systems able to adapt

* Corresponding author. Tel.: +30 216 5000 500; fax: +30 216 5000 599.
E-mail address: azafeiropoulos@ubitech.eu

Peer-review under responsibility of organizing committee of the international conference on cloud forward: From Distributed to Complete Computing
doi:10.1016/j.procs.2016.08.288

to their operational environment conditions is required; this is highly associated with the transition from monolithic applications to applications consisting of a set of microservices. The term microservice has been lately introduced referring to the design and development of software that is independently deployable and orchestratable. For instance, a microservice can be part of an application and able to scale independently of the rest application parts.

Based on the transition to applications consisting of microservices, applications are denoted in the form of service graphs. Such service graphs are then deployable and orchestratable over a programmable infrastructure. By programmable, we refer to a virtualized infrastructure that can be dynamically managed in order to accommodate the deployment requests. Thus, a microservices-based approach can be considered as an approach that incorporates DevOps practices. This creates the need for the design, development and deployment of orchestration frameworks able to manage the deployment and operation of distributed applications consisting of microservices.

In the ARCADIA project[2], a microservices-based software development paradigm and orchestration framework for *reconfigurable-by-design distributed applications* is being developed. The framework is based on the definition of a representation context model, the design and development of an orchestrator for deploying and managing distributed applications and the design and development of a policies management framework for real-time policies enforcement over the deployed applications. The designed context model and the associated policy management framework are presented in brief in Sections 2, 3 respectively while in Section 4 we conclude the paper.

2. ARCADIA Context Model

A *distributed reconfigurable-by-design application* (ARCADIA app from now on) is defined as a multi-tier cloud application consisting of application components chained among them as well as with other software entities illustrating network functions. The definition *reconfigurable-by-design* implies that a distributed application is designed to be context-aware, able to adapt its components to the context and reconfigure its structure accordingly. Such an application may expand or shrink by supporting horizontal scaling (out and in) of each software component in the service chain while it may reform (change its structure) by including or excluding software components from the chain and/or change routing among them as needed.

An ARCADIA app runs over a *programmable infrastructure*. An orchestrator is being developed -called Smart Controller- for supporting optimal placement and management of the apps over the programmable infrastructure (Figure 1a)[3]. Through programming interfaces, the infrastructure enables configuration and control of its computing, network and storage resources along with flexible illustration of network functions according to the apps needs.

The ARCADIA context model[4] aims at representing the development, deployment and orchestration aspects of ARCADIA apps. It is used in all service lifecycle phases (i.e. development, composition, deployment planning, execution) in order to conceptualize specific aspects of distributed applications that are essential by the ARCADIA architectural components. The context model has four facets; the Component model, the Service graph model, the Service deployment model and the Service runtime model.

The *ARCADIA Component Model (CM)* represents the most granular executable unit of an ARCADIA app that is a software component (microservice). Several CM instances can be combined towards the specification of a service graph. Distributed applications are practically an instantiation of a complex service graph while Components can be considered as the building blocks of these graphs. The CM schema consists of a set of elements, including the 'ComponentConfiguration' element that encapsulates information regarding the available configuration aspects of the CM instance which is processed during the deployment of the Component, the 'Requirements' element which encapsulates the set of requirements that have to be met for the smooth execution of the CM instance (e.g. CPU speed, Memory), the 'ExposedInterfaces' element which describes the set of exposed interfaces that are the most granular exposable functions of the Component Model instance, the 'RequiredInterfaces' element that describes the set of required interfaces which are the most granular consumable functions required by the CM instance and the 'CoreHooks' element that includes the set of mandatory lifecycle Component management hooks.

Many ARCADIA CMs can be combined in order to create an *ARCADIA Service Graph Model (SGM)* which is practically a directed graph (DG) containing three main elements; the 'GraphNode Descriptor' element that encapsulates information regarding the graph nodes of the DG, the 'VirtualLinkDescriptor' element that provides information related to the links of the DG and the 'GraphMonitoringDescriptor' element that encapsulates information regarding monitoring metrics of the entire graph (e.g. end-to-end delay in the service provision).

Once a Service Graph instance is created, any potential service provider can use ARCADIA in order to instantiate this graph (i.e. the complex service that is represented by this graph). Instantiation refers to the selection/allocation of a specific resource that has to be decided according to a specific high level policy. When the desired placement of each component is decided, a specific schema should track the association of the components placement to the IaaS resources. This is the role of the *Service Deployment Model (SDM)*. The SDM is agnostic to the placement process. Thus, it is irrelevant how a placement decision was taken; it could have been a manual process. An SDM consists of a 'ServiceGraphModelReference' element that encapsulates the reference graph used for a specific deployment, a 'ConfigurationDescriptor' element that encapsulates the information for the selected configuration for each Component in the Service Graph and the 'ComponentPlacementDescriptor' that encapsulates the information for the multi-IaaS Placement of each Component in the Service Graph (e.g. IaaS connectivity aspects, placement constraints, execution environment constraints).

An *ARCADIA Service Runtime Model (SRM)* represents, conceptually, an instance of a deployed Service Graph which follows the 'rules' imposed by the orchestrator (Smart Controller)[3]. These rules are 'serialized' in the SDM. Therefore, in a non-conceptual way, an SRM is an extension of a Deployment Model. The SRM consists of the 'DeploymentModel' element that encapsulates an entire Deployment Model instance and the 'RuntimeBindings' element that encapsulates runtime information of all running Component instances.

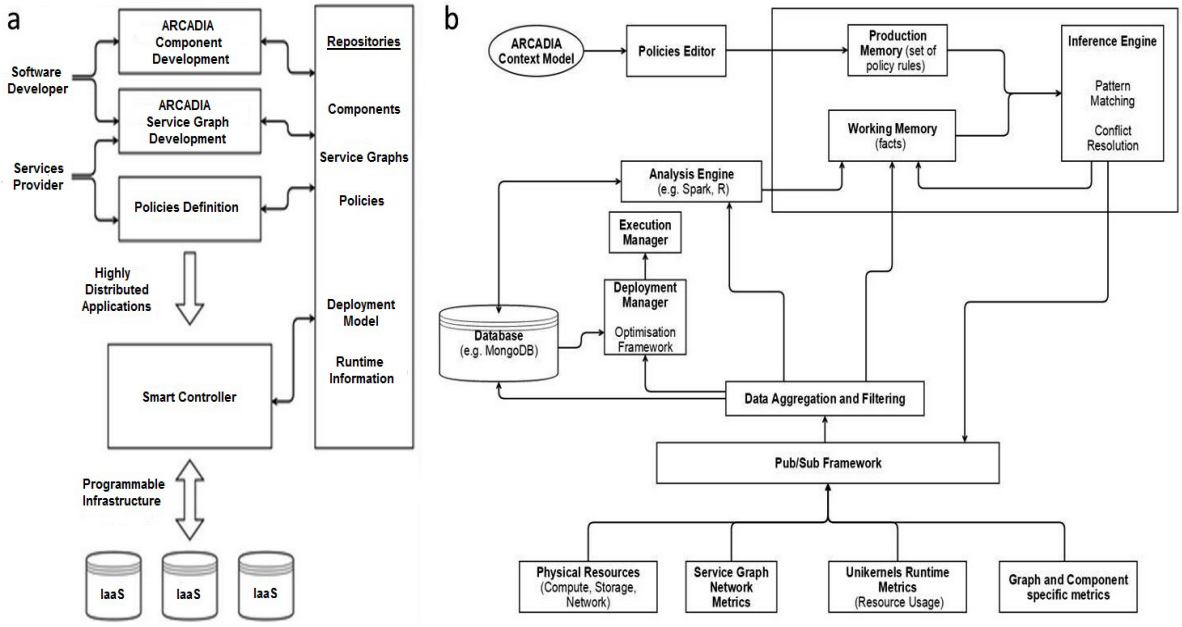

Figure 1. (a) ARCADIA High Level Architectural View; (b) ARCADIA Policies Management Framework

3. Policies Management Framework

The Policies Management Framework (PMF) in ARCADIA (Figure 1b) is providing policies enforcement over the deployed service graphs following a continuous *match-resolve-act approach*. Specifically, the *match phase* regards the mapping of the set of applied rules which are satisfied based on the data streams coming from the monitoring infrastructure, the *resolve phase* regards the process of conflict resolution -if any- among the satisfied rules taking into account the pre-defined salience of each rule, while the *act phase* regards the provision of a set of suggested actions by the PMF to the orchestration components; the Deployment Manager and the Execution Manager of the Smart Controller, responsible for service graphs placement and management respectively. *Policies enforcement* is realized through a rule-based framework that attempts to derive execution instructions based on the current set of data and the *active rules*; rules associated with the deployed service graphs over at each point of time.

The PMF consists of (i) the *working memory (WM)*; facts based on the provided data, (ii) the *production memory (PM)*; set of defined rules, and (iii) an *inference engine (IE)* that supports reasoning and conflict resolution over the provided set of facts and rules as well as triggering of the appropriate actions. Data is fed to the WM through the *Data Aggregation and Filtering component* that is responsible to collect data based on a set of active monitoring probes, as well as to support a set of data management operations (e.g. calculation of average values in specific time windows). The PM is also fed by policies associated with the deployed service graphs, as provided through the *Policies Editor* - the editor made available to service providers for policies definition.

Data monitoring and management processes are supported through a set of *active and passive monitoring probes*. Such probes collect information regarding availability and usage of physical resources (compute, storage and network resources) over a multi-IaaS infrastructure, information regarding resources usage per deployed component/microservice monitoring APIs as well as information regarding custom metrics of the deployed service graphs and/or components, as defined on behalf of the software developer. PMF dynamically handles and converts the collected data to WM facts. Such facts can then be matched with already defined rules on the active policies.

Definition of rules per policy is supported through the *Policy Editor* in a per service graph basis, based on the concepts represented in the *Context model*. A service graph may be associated with a set of policies, however only one can be active during its deployment and execution time. Each policy is consisted by a set of rules. Each rule consists of the expressions part -denoting a set of conditions to be met- and the actions part -denoting actions upon the fulfillment of the conditions. The defined policies are translated to a set of rules that become part of the PMF PM. Expressions may regard custom metrics of a service graph or a component/microservice. An indicative expression is as follows: “if microserviceX.avg_cpu_usage is greater than 80%” and can be combined in various ways (and/or/grouping/subgrouping) with other expressions.

The potential actions of a policy are classified in four categories that each one regards to: (i) the components/microservices lifecycle management (e.g. start, stop, destroy), (ii) the management of mutable components/microservices configuration parameters (e.g. microserviceX.maxConnections set to 100), (iii) the management of horizontal or vertical scaling functionalities per component/microservice, and (iv) the IaaS management functions such as adding or removing resources for specific components/microservices.

Each rule has attached a specific *salience* that is used as a priority indicator during *conflict resolution* by the IE. A time window is specified per rule for the examination of the provided expressions and the support of inference functionalities during that. When attaching a specific runtime policy to a grounded service graph, the specified set of policy rules are deployed to the PMF PM, while the WM agent is constantly feeding the WM with new facts.

4. Conclusions

Taking into account the transition to microservices based software development paradigms, in the current manuscript, a novel approach is presented which aims at facilitating the development and policy-aware placement and management of reconfigurable-by-design distributed applications over programmable infrastructure.

Acknowledgements

This work has been co-funded by the ARCADIA project, a European Commission research program under Contract Number H2020-645372.

References

1. The Reactive Manifesto, Available Online: http://www.reactivemanifesto.org/
2. The ARCADIA Horizon2020 Project. Available Online: http://arcadia-framework.eu/
3. ARCADIA H2020 Project, D2.3 - Description of the ARCADIA Framework, Available Online: http://www.arcadia-framework.eu/wp/documentation/deliverables/
4. ARCADIA Horizon2020 Project, D2.2 – Definition of the ARCADIA Context Model, Available Online: http://www.arcadia-framework.eu/wp/documentation/deliverables/